METHOD OF GENERATING PACKETS
WITHOUT REPETITION IN VERIFICATION
OF A DEVICE

5   Lee-Yin Chee

Kheng Guan (Nigel) Tan

Sie-Boo Chiang

BACKGROUND OF THE INVENTION

10

1.   Technical Field

This invention relates generally to device verification, and more particularly, to a method wherein testing combinations are generated without repetition in such verification.

15   ### 2.   Background Art

In the verification of an Ethernet Network Controller, the data packet format for IPSec (Internet Protocol Security) processing typically includes three parts, i.e., the MAC (Media Access Control) header, IP (Internet Protocol) Datagram, and FCS (Frame Check Sequence). For the MAC header, typically, only six different combinations of packet format are involved

20   and need to be tested. However, for the IP Datagram, in the case of the IPv6 header with IPv6 extension headers, several thousand different combinations are involved.

In such verification, a main strategy is the generation of random simulation vectors. In order to fulfill this strategy, a vector generation model is to generate random packet combinations. It will be understood that each simulation is to cover each and every legal packet

25   class. It will also be noted that there are many legal packet classes which can traverse through the network.

However, with each packet being randomly generated, a packet for a particular packet class will be very likely be generated more than once before all legal packet classes are covered, resulting in the device being tested for the same packet class more than once, possibly several

30   times. It will be understood that additional testing for that packet class (after the first test based on that packet class) is inefficient, unnecessary and time-consuming.

Therefore, what is needed is a method of avoiding repetition in the generation of combination values used in testing a device.

35

DISCLOSURE OF THE INVENTION

In the present method for use in verification of a device, a plurality of injection flags are provided, each of which is associated with one of a plurality of packet classes. Each injection flag may be of a first or a second state. Next, a packet is generated. If the injection flag of the packet class of the generated packet is in the second state, it is indicated that a packet of that packet class has already been generated, and the device is not tested. If the injection flag of the packet class of the generated packet is in the first state, the device is tested and the injection flag of the packet class of the generated packet is set to the second state.

The present invention is better understood upon consideration of the detailed description below, in conjunction with the accompanying drawings. As will become readily apparent to those skilled in the art from the following description, there is shown and described an embodiment of this invention simply by way of the illustration of the best mode to carry out the invention. As will be realized, the invention is capable of other embodiments and its several details are capable of modifications and various obvious aspects, all without departing from the scope of the invention. Accordingly, the drawings and detailed description will be regarded as illustrative in nature and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as said preferred mode of use, and further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a flow diagram illustrating the present invention.

BEST MODE(S) FOR CARRYING OUT THE INVENTION

Reference is now made in detail to a specific embodiment of the present invention which illustrates the best mode presently contemplated by the inventors for practicing the invention.

Figure 1 illustrates the steps of the present invention. In the present simulation, initially, each legal packet class is provided with a 1-bit injection flag. These injection flags will form a long sequence of binary encoding, which may be called network_injection_state. In a simple example:

network_injection_state = { class0, class1, class2, class3 }

the injection flags are all initially in the state 0, for example,

5   Initial State: 4'b0000

Next, let us assume that the packet generator (randomly) generates a class2 packet. If the injection flag of the packet class of the generated class2 packet = 0 (it will be remembered that all injection flags were initially set to 0), the test is run, and the injection flag of that packet class

10 is set to 1. This results in the state of the injection flags

    4'b0010

Next, assume that the packet generator randomly generates a class3 packet. If the

15 injection flag of the packet class of the generated class3 packet = 0 (as indeed is the case), the test is run, and the injection flag of that packet class is set to 1. This results in the state of the injection flags

    4'b0011

20

Next, assume that the package generator randomly again generates a class3 packet. In this situation, the injection flag of the packet class of the generated class3 packet has previously been set to 1. In this situation, the generated packet class3 packet is discarded, and another packet is randomly generated. Similar to the above, for any generated packet, it will be noted

25 whether the injection flag of the packet class thereof is in a 0 or a 1 state. A 0 state indicates that a packet of that packet class had not yet been generated, and the test will be run. A 1 state indicates that a packet of that packet class has been generated, and a test will not be run. The steps described above and illustrated in Figure 1 are repeated until all the injection flags have been set to 1:

30

    4'b1111

This indicates that packets of all packet classes have been generated, and the simulation is ended.

35   The overall process can be described in accordance with Figure 1. Initially, each legal packet class is provided with a 1-bit injection flag, with each value thereof being sent to 0 (10).

Next, a packet is randomly generated (12). The state of the injection flag of the packet class of the generated packet is noted (14). If the state of the injection flag is 0, the test is run (16), and the injection flag of bad packet class is set to 1 (18). If the state of the injection flag is 1, a test is not run, the generated packet is discarded, another packet is randomly generated, and the

5   procedure subsequent thereto as described above is repeated. The process is continued until all such injection flags have been set to 1. When all the injection flags have been set to 1 (20), the simulation is ended (22).

As noted, during each simulation run, only one packet from each legal packet class will be generated to test the controller. The problem of repetition and inefficiency described above is

10   thus avoided, and simulation time is greatly reduced. It will also be noted that for any given simulation, the sequence of packets is a random, so that each simulation run will be different from the others.

The foregoing description of the embodiment of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the

15   invention to the precise form disclosed. Other modifications or variations are possible in light of the above teachings.

The embodiment was chosen and described to provide the best illustration of the principles of the invention and its practical application to thereby enable one of ordinary skill of the art to utilize the invention in various embodiments and with various modifications as are

20   suited to the particular use contemplated. All such modifications and variations are within the scope of the invention as determined by the appended claims when interpreted in accordance with the breadth to which they are fairly, legally and equitably entitled.